Achieving Fully Proportional Representation is Easy in Practice

Piotr Skowron
University of Warsaw
Warsaw, Poland
p.skowron@mimuw.edu.pl

Piotr Faliszewski
AGH University of Science and Technology
Krakow, Poland
faliszew@agh.edu.pl

Arkadii Slinko
University of Auckland
Auckland, New Zealand
a.slinko@auckland.ac.nz

ABSTRACT

We provide experimental evaluation of a number of known and new algorithms for approximate computation of Monroe's and Chamberlin-Courant's rules. Our experiments, conducted both on real-life preference-aggregation data and on synthetic data, show that even very simple and fast algorithms can in many cases find near-perfect solutions. Our results confirm and complement very recent theoretical analysis of Skowron et al., who have shown good lower bounds on the quality of (some of) the algorithms that we study.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

Keywords

algorithms, parliamentary elections, winner-determination

1. INTRODUCTION

Many countries are governed using indirect democracy, where the people do not make decisions directly, but rather select representatives (e.g., a parliament, a senate, a congress) who rule in their interest. Unfortunately, relatively little effort was so far invested in the algorithmic study of procedures for electing committees of representatives (few exceptions include papers [2,5,9,14]). Here, we consider two particularly appealing rules for electing a set of representatives, namely those of Monroe and of Chamberlin and Courant, and we argue that while these rules in the worst case scenario may be difficult to compute [5,14], in practice, very simple and efficient algorithms find almost-perfect approximate results.

There are several ways in which countries can choose their parliaments (or, more generally, in which societies can choose committees of representatives). Often, voters are divided into districts and in each district we hold a local election. For the case of single-representative districts, in each district we have a single-winner election held according to one of the standard, well-known, rules such as the Plurality rule or Borda's rule. In particular, if the Plurality rule is used then this method is known as First-Past-the-Post (FPP): In each district the candidate supported by the largest number of voters is elected. However, FPP has a number of drawbacks. For example, it is possible that in a country with two major parties, A and B, even if 49% of the citizens support party B, only members of party A enter the parliament (this happens if in each district party A has a slight advantage over party B). Indeed, under FPP the election organizers are particularly tempted to tamper with the partition of voters into districts. To circumvent this problem one might use multi-representative districts, where elections are held using some multi-winner voting rule (e.g., using Single Transferable Vote (STV),[1] or using a voting rule that assigns scores to the candidates and picks a group of those with highest scores). However, this approach only partially solves the problem. Further, compared to single-representative districts, multi-representative districts loosen the connection between the candidates and the voters that have elected them.

Fortunately, there is a very attractive way to avoid the problems mentioned above: Instead of using fixed districts, we may partition the voters dynamically, based on the votes that they cast. Indeed, this is exactly the idea behind the fully proportional representation rules of Monroe [10] and of Chamberlin and Courant [3]. If we seek a parliament of K representatives, then Monroe's rule says that we should pick a set of K candidates for whom there is an assignment of these candidates to the voters such that: (a) each candidate is assigned to roughly the same number of voters, and (b) the total satisfaction of the voters (measured in one of the ways introduced later) is maximal. Chamberlin-Courant's rule is similar except that it allows each selected candidate to be matched to a different number of voters. (Thus if one were to elect a parliament using Chamberlin-Courant's rule then one should use weighted voting within the parliament, weighted by the number of voters matched to each representative.)

In the above description we focus on political elections, but we mention that both Monroe's rule and Chamberlin-Courant's rule have many different applications as well. For example, Skowron et al. [15] have presented several (multi-agent) resource allocation settings that can be modeled using these rules and Lu and Boutilier [5] have proposed to use Chamberlin-Courant's rule for constructing recommendations for groups of agents.

Unfortunately, computing Monroe's and Chamberlin-Courant's rules is both NP-hard [5,14] and difficult in the parametrized sense [2]. Thus using them in practice might simply be impossible. The goal of this paper is to show that not all is lost. We provide experimental evaluation of a number of known and new algorithms for approximate computation of Monroe's and Chamberlin-Courant's rules. Our experiments, conducted both on real-life preference-aggregation data and on synthetic data, show that even

[1] STV for more than one winner is sometimes referred to as "Alternative Vote" (AV). In a recent referendum Great Britain rejected AV as a method for choosing its members of parliament.

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

very simple and fast algorithms can in many cases find near-perfect solutions. Our results confirm and complement very recent theoretical analysis of Skowron et al. [15], who have shown good lower bounds on the quality of (some of) our algorithms. While for single-winner rules using approximate algorithms may be debatable, for the case of electing a large body of representatives, e.g., a parliament, using approximation algorithms seems far easier to justify. Indeed, a good approximate solution for Monroe's or Chamberlin-Courant's rule represents the society almost as well as a perfect solution would.

The paper is organized as follows. In Section 2 we formally define Monroe's and Chamberlin-Courant's rules. In Section 3 we give an overview of the algorithms that we evaluate and in Section 4 we describe the data sets that we use in our experiments. Section 5 contains our main results. We conclude in Section 6.

2. PRELIMINARIES

In this section we briefly review basic notions regarding social choice theory and we define Monroe's [10] and Chamberlin and Courant's [3] fully proportional representation rules. We assume the reader is familiar with standard notions regarding algorithms. For each positive integer n, by $[n]$ we mean the set $\{1, \ldots, n\}$.

Elections. We consider elections over a given set $A = \{a_1, \ldots, a_m\}$ of alternatives. We have a set $N = [n]$ of agents (the voters), where each voter i, $1 \leq i \leq n$, has a preference order $\succ_i$ over A. A preference order of an agent i is a linear order over the set A; the maximal element is this agent's most preferred alternative, the minimal element is this agent's least preferred alternative, and the alternatives in the middle represent the agent's spectrum of preference. We refer to the collection $V = (\succ_1, \ldots, \succ_n)$ as the preference profile for a given election.

Let us fix an agent i, $1 \leq i \leq n$, and an alternative $a \in A$. By $\mathrm{pos}_i(a)$ we mean the position a has in i's preference order. If a is i's most preferred candidate then $\mathrm{pos}_i(a) = 1$, and if a is i's least preferred candidate then $\mathrm{pos}_i(a) = \|A\| = m$.

Positional Scoring Functions. Let m be the number of candidates in eleciton. A *positional scoring function (PSF)* is any function $\alpha\colon [m] \to \mathbb{N}$ that satisfies the following two conditions: (a) $\alpha(m) = 0$, and (b) for each i, j, $1 \leq i < j \leq m$, $\alpha(i) \geq \alpha(j)$. In Monroe's and in Chamberlin-Courant's proportional representation rules we will match agents to the alternatives that represent them. Intuitively, $\alpha(i)$ is the amount of satsifaction that an agent derives from being represented by an alternative that this agent ranks on the i'th position. In this paper we focus on Borda count PSF, which for m alternatives is defined as $\alpha^m_{\mathrm{Borda}}(i) = m - i$. However, occasionally we will consider other PSFs as well.

In our algorithms we assume that the PSF α to be used is given explicitly, as a vector $(\alpha_1, \ldots, \alpha_m)$ of integers such that for each i, $1 \leq i \leq m$, $\alpha(i) = \alpha_i$. We will implcitly assume that the number of alternatives matches the domain of the given PSF.

Proportional Representation. Let $A = \{a_1, \ldots, a_m\}$ be the set of alternatives and $N = [n]$ be the set of agents (with preference orders over A). A representation function is any function $\Phi\colon N \to A$. For an m-candidate PSF α and a representation function Φ, Φ's satisfaction is defined as:

$$\alpha(\Phi) = \sum_{i=1}^{n} \alpha(\mathrm{pos}_i(\Phi(i))).$$

Let K be a positive integer. A K-CC-representation function is any representation function Φ such that $\|\Phi^{-1}(N)\| \leq K$ (that is, any representation function that matches voters to at most K alternatives). A K-Monroe-representation function Φ is any K-CC-representation function that additionally satisfies the following requirement: For each $a \in A$ it holds that either $\lfloor \frac{n}{K} \rfloor \leq \|\Phi^{-1}(a)\| \leq \lceil \frac{n}{K} \rceil$ or $\|\Phi^{-1}(a)\| = 0$ (that is, each alternative represents either roughly $\frac{n}{K}$ agents or none of them).

We will also consider partial representation functions. A partial CC-representation function is defined in the same way as a regular one, except that it may assign a null alternative, $\perp$, to some of the agents. By convention, we take that for each agent i we have $\mathrm{pos}_i(\perp) = m$. A partial Monroe-representation function is defined analogously: It may assign the null alternative to some voters (there are no constraints on the number of agents to whom the null alternative is assigned) but it must be possible to extend it to a regular Monroe-representation function by replacing the occurrences of the null alternative with the real ones.

We now define Monroe's and Chamberlin-Courant's (CC) rules.

DEFINITION 1. *Let R be a member of $\{\mathrm{Monroe}, \mathrm{CC}\}$. Let $A = \{a_1, \ldots, a_m\}$ be a set of alternatives, $N = [n]$ be a set of agents, and α be an m-candidate PSF. Let K be the size of the set of representatives that we seek ($K \leq m$). We say that a K-element set W, $W \subseteq A$, is a set of α-R winners if there exists a K-R-representation function $\Phi : N \to W$ such that for every other K-R-representation function Ψ it holds that $\alpha(\Phi) \geq \alpha(\Psi)$.*

We point out that for both Monroe's and Chamberlin-Courant's rule there may be several different winner sets and that some form of tie-breaking should be applied in these settings. Here we disregard tie-breaking and simply are interested in *some* winner set (and, not being able to compute that, in any set with as high a satsifaction as possible).

It is well-known that for many natural families of PSFs, both for Monroe's rule and for Chamberlin-Courant's rule, it is NP-complete to decide if there exists a winner set that achieves a given satisfaction [2,5,14]. However, for each R in $\{\mathrm{Monroe}, \mathrm{CC}\}$, for each PSF α (with the domain matching the number of alternatives in the election), and for each set S of up to K alternatives we can compute in polynomial time a (possibly partial) K-R-representation function Φ^S_R that maximizes the agent satisfaction under the condition that agents are matched to the alternatives in S only. Indeed, it is easy to see that for α-CC this function is:

$$\Phi^S_{\mathrm{CC}}(i) = \mathrm{argmin}_{a \in S} \mathrm{pos}_i(a)$$

and that it is never a partial representation function. For the case of α-Monroe, computing Φ^S_{Monroe} is more involved and requires solving a certain min-cost/max-flow problem (see the work of Betzler et al. [2]; here if $\|S\| < K$ then Φ^S_{Monroe} is a partial Monroe-representation function). One can see that for a given set S, there may be many different (partial) K-Monroe-representation functions that achieve optimal satisfaction; when we write Φ^S_{Monroe}, we mean, w.l.o.g., the particular one computed by the algorithm of Betzler et al. [2].

3. ALGORITHMS

Let us now describe the algorithms that we will consider in this work. Some of our algorihtms can be applied both to Monroe's rule and to Chamberlin-Courtant's rule, while some are specific to only one of them. For each algorithm we will exactly specify for which rules it is applicable and, if it is applicable to both, what are the differences.

While most of the algorithms described below are based on ones already given in the literature, in a number of cases we added heuristics on top of existing algorithms (which proved to be quite

effective, as we will see later) and, in one case, provided a completely new theortical analysis. For each algorithm we will carefully describe what was already known in the literature, and which additions are due to this paper.

Throughout this section we assume we are given the following setting. $A = \{a_1, \ldots, a_m\}$ is a set of alternatives, α is an m-candidate PSF, $N = [n]$ is a set of agents, each with a preference order over A, and K is a positive integer, $K \leq m$ (the size of the committee we want to elect).

3.1 ILP Formulation (Monroe and CC)

To measure the quality of our approximation algorithms, we compare their results against optimal solutions that we obtain using integer linear programs (ILPs) that describe Monroe's and Chamberlin-Courant's rules. An ILP for Chamberlin-Courant's rule, for arbitraty PSF α, was provided by Lu and Boutilier [5]; the analogous formulation for Monroe's rule was provided by Potthoff and Brams [13]. We used the GLPK 4.47 package (GNU Linear Programming Kit, version 4.47) to solve these ILPs, whenever it was possible to do so in reasonable time.

3.2 Algorithms A, B, and C (Monroe)

Skowron et al. [15] have suggested and studied the following algorithm for Monroe's rule, which we will call Algorithm A. We start with an empty partial Monroe-representation function Φ and we execute K iterations. In each iteration we do the following:

1. For each alternative $a \in A$ that does not yet represent any agents, we compute the maximal satisfaction that some not-yet-represented $\lceil \frac{n}{K} \rceil$ agents derive from being represented by a (we call this number $\mathrm{score}(a)$ and we refer to these agents as $\mathrm{bests}(a)$).
2. We pick an alternative a with maximum $\mathrm{score}(a)$ and extend Φ by assigning a to represent agents in $\mathrm{bests}(a)$.

This algorithm clearly works in polynomial time. Skowron et al. [15] have shown that for $\alpha^m_{\mathrm{Borda}}$ it finds a solution whose satisfaction is at least a $(1 - \frac{K-1}{2(m-1)} - \frac{H_K}{K})$ fraction of a (possibly nonexistent) perfect solution, where each agent is represented by his or her top preference (H_K is the K'th harmonic number, i.e., $H_K = \sum_{i=1} \frac{1}{i} = \Theta(\log K)$). This suggests that the algorithm performs best in elections where the size of the committee we seek is relatively small with respect to the number of alternatives.

Based on Algorithm A we have derived Algorithm B. The only difference is that after completing the operation of Algorithm A, we take the set S of alternatives that were assigned to represent some agents by Algorithm A, and replace function Φ with function Φ^S_{Monroe}, that optimally reassigns the alternatives to the voters. This very simple heuristic turned out to noticeably improve the results of the algorithm in practice (and, of course, the approximation guarantees carry over from Algorithm A to Algorithm B).

Algorithm C is a further heuristic improvement over Algorithm B. This time the idea is that instead of keeping only one partial function Φ, we keep a list of up to d partial representation functions, where d is a parameter of the algorithm. At each iteration, given these d partial representation functions, for each Φ of them and for each alternative a that does not yet have agents assigned to by this Φ, we compute an optimal extension of this Φ that assigns agents to a. As a result we obtain possibly more than d (partial) representation functions. For the next iteration we keep those d of them that have highest satisfaction.

We provide pseudocode for Algorithm C in Figure 1. If we take $d = 1$, we obtain Algorithm B. If we also disregard the last two lines prior to returning solution, we obtain Algorithm A.

Figure 1: The pseudocode for Algorithm C.

Notation: $\Phi \leftarrow$ a map defining a (partial) representation function, iteratively built by the algorithm.
$\Phi^{\leftarrow} \leftarrow$ the set of agents already represented by some alternative
$\Phi^{\rightarrow} \leftarrow$ the set of alternatives already used in the representation function.
$Par \leftarrow$ a list of partial representation functions

$Par = []$
Par.push($\{\}$)
for $i \leftarrow 1$ **to** K **do**
  $newPar = []$
  for $\Phi \in Par$ **do**
    $score \leftarrow \{\}$
    $bests \leftarrow \{\}$
    foreach $a_i \in A \setminus \Phi^{\rightarrow}$ **do**
      $agents \leftarrow$ sort $N \setminus \Phi^{\leftarrow}$ so that $j \prec k$ in $agents$ $\implies pos_j(a_i) \leq pos_k(a_i)$
      $bests[a_i] \leftarrow$ chose first $\lceil \frac{N}{K} \rceil$ elements of $agents$
      $\Phi' \leftarrow \Phi$
      foreach $j \in bests[a_i]$ **do**
        $\Phi'[j] \leftarrow a_i$
      $newPar$.push(Φ')
    sort $newPar$ according to descending order of the total satisfaction of the assigned agents
    $Par \leftarrow$ chose first d elements of $newPar$
for $\Phi \in Par$ **do**
  $\Phi \leftarrow$ compute the optimal representative function using an algorithm of Betzler et al. [2] for the set of winners $\Phi^{\rightarrow}$
return *the best representative function from* Par

Figure 2: Pseudocode for Algorithm GM.

Notation: R is either Monroe or CC.
$S \leftarrow \emptyset$
for $i \leftarrow 1$ **to** K **do**
  $a \leftarrow \mathrm{argmax}_{a \in A \setminus S} \alpha(\Phi_R^{S \cup \{a\}})$
  $S \leftarrow S \cup \{a\}$
return Φ^S_M

3.3 Algorithm GM (Monroe and CC)

Algorithm GM (greedy marginal improvement) was introduced by Lu and Boutilier for the case of Chamberlin-Courant's rule. Here we generalize it to apply to Monroe's rule as well, and we show that it is a $1 - \frac{1}{e}$ approximation algorithm for α-Monroe. We point out that this is the first approximation result for Monroe rule that applies to all PSFs α (approximability results of Lu and Boutilier [5] did not apply to α-Monroe, and results of Skowron et al. [15] applied to Monroe with Borda count PSF only). For Monroe's rule, the algorithm can be viewed as extending Algorithm B.

Let R be one of Monroe and CC. The algorithm proceeds as follows. We start with an emtpy set S. Then we execute K iterations. In each iteration we find an alternative a that is not assigned to agents yet, and maximizes the value $\Phi_R^{S \cup \{a\}}$. (A certain disadvantage of this algorithm for the case of Monroe is that it requires a large number of computations of Φ^S_{Monroe}, which is a slow process based on min-cost/max-flow algorithm.) We provide the pseudocode for Algorithm GM in Figure 2.

THEOREM 1. *Algorithm GM is an $(1-1/e)$-approximation algorithm for the Monroe'e election problem for arbitrary positional scoring functions.*

PROOF. The proof is based on the powerful result of Nemhauser et al. [11], who have shown that greedy algorithms achieve $1 - \frac{1}{e}$ approximation ratio when used to optimize submodular functions. Let A be a set of alternatives, $N = [n]$ a set of agents with prefer-

ences over A, α an $\|A\|$-candidate PSF, and $K \leq \|A\|$ the number of representatives that we want to elect.

We consider function $z : 2^A \to \mathbb{N}$ defined, for each set S, $S \subseteq A$ and $\|S\| \leq K$, as $z(S) = \alpha(\Phi^S_{\text{Monroe}})$. Clearly, $z(S)$ is monotonic (that is, for each two sets A and B, if $A \subseteq B$ and $\|B\| \leq K$ then $z(A) \leq z(B)$. The main part of the proof below is to show that z is submodular (we provide the definition below).

Since $\text{argmax}_{S \subset A, \|S\|=K} z(S)$ is the set of winners of our election (under α-Monroe) and since Algorithm GM builds the solution iteratively by greedily extending initially empty set S so that each iteration increases the value of $z(S)$ maximally, by the results of Nemhauser et al. [11] we get that Algorithm GM is a $(1 - \frac{1}{e})$-approximation algorithm.

Let us now prove that z is submodular. That is, our goal is to show that for each two sets S and T, $S \subset T$, and each alternative $a \notin T$ it holds that $z(S \cup \{a\}) - z(S) \geq z(T \cup \{a\}) - z(T)$. First, we introduce a notion that generalizes the notion of a partial set of winners S. Let $s : A \to \mathbb{N}$ denote a function that assigns a capacity to each alternative (i.e., s gives a bound on the number of agents that a given alternative can represent). Intuitively, each set S, $S \subseteq A$, corresponds to the capacity function that assigns $\lceil \frac{n}{k} \rceil$ to each alternative $a \in S$ and 0 to each $a \notin S$. Given a capacity function s, we define a partial solution Φ^s_{Monroe} to be one that maximizes the total satisfaction of the agents and that satisfies the capacity constraints: $\forall_{a \in S} \|(\Phi^s_{\text{Monroe}})^{-1}(a)\| \leq s(a)$. To simplify notation, we write $s \cup \{a\}$ to denote the function such that $(s \cup \{a\})(a) = s(a) + 1$ and $\forall_{a' \in S}(s \cup \{a\})(a') = s(a')$. (Analogously, we interpret $s \setminus \{a\}$ as subtracting one from the capacity for a; provided it is nonzero.) Also, by $s \leq t$ we mean that $\forall_{a \in A} s(a) \leq t(a)$. We extend our function z to allow us to consider a subset of the agents only. For each subset N' of the agents and each capacity function s, we define $z_{N'}(s)$ to be the satisfaction of the agents in N' obtained under Φ^s_{Monroe}. We will now prove a stronger variant of submodularity for our extended z. That is, we will show that for each two capacity functions s and t it holds that:

$$s \leq t \Rightarrow z_N(s \cup \{a\}) - z_N(s) \geq z_N(t \cup \{a\}) - z_N(t) \quad (1)$$

Our proof is by induction on N. Clearly, Equation (1) holds for $N' = \emptyset$. Now, assuming that Equation (1) holds for every $N' \subset N$ we will prove its correctness for N. Let i denote an agent such that $\Phi^{t \cup \{a\}}_{\text{Monroe}}(i) = a$ (if there is no such agent then clearly the equation holds). Let $a_s = \Phi^s_{\text{Monroe}}(i)$ and $a_t = \Phi^t_{\text{Monroe}}(i)$. We have:

$$\begin{aligned} z_N(t \cup \{a\}) - z_N(t) = {} & \alpha(\text{pos}_i(a)) + z_{N \setminus \{i\}}(t) \\ & - \alpha(\text{pos}_i(a_t)) - z_{N \setminus \{i\}}(t \setminus \{a_t\}). \end{aligned}$$

We also have:

$$\begin{aligned} z_N(s \cup \{a\}) - z_N(s) \geq {} & \alpha(\text{pos}_i(a)) + z_{N \setminus \{i\}}(s) \\ & - \alpha(\text{pos}_i(a_s)) - z_{N \setminus \{i\}}(s \setminus \{a_s\}). \end{aligned}$$

Since Φ^t_{Monroe} describes an optimal representation function under the capacity restrictions t, we have that:

$$\alpha(\text{pos}_i(a_t)) + z_{N \setminus \{i\}}(t \setminus a_t) \geq \alpha(\text{pos}_i(a_s)) + z_{N \setminus \{i\}}(t \setminus \{a_s\})$$

Finally, from the inductive hypothesis for $N' = N \setminus \{i\}$ we have:

$$z_{N \setminus \{i\}}(s) - z_{N \setminus \{i\}}(s \setminus \{a_s\}) \geq z_{N \setminus \{i\}}(t) - z_{N \setminus \{i\}}(t \setminus \{a_s\})$$

Figure 3: Pseudocode for Algorithm P.

Notation: We use the same notation as in Algorithm 1; $\text{num\_pos}_x(a) \leftarrow \|\{i \in [n] \setminus \Phi^{\leftarrow} : pos_i(a) \leq x\}\|$ (the number of not-yet assigned agents that rank alternative a in one of their first x positions)

```
Φ = {}
x = ⌈mW(K)/K⌉
for i ← 1 to K do
    a_i ← argmax_{a∈A\Φ→} num_pos_x(a)
    foreach j ∈ [n] \ Φ← do
        if pos_j(a_i) < x then
            Φ[j] ← a_i
foreach j ∈ A \ Φ← do
    a ← such server from Φ→ that ∀_{a'∈Φ→} pos_j(a) ≤ pos_j(a')
    Φ[j] ← a
```

By combining these inequalities we get:

$$\begin{aligned} z_N(s \cup \{a\}) - z_N(s) &\geq \alpha(\text{pos}_i(a)) + z_{N \setminus \{i\}}(s) \\ &\quad - (\alpha(\text{pos}_i(a_s)) + z_{N \setminus \{i\}}(s \setminus \{a_s\})) \\ &\geq \alpha(\text{pos}_i(a)) - \alpha(\text{pos}_i(a_s)) \\ &\quad + z_{N \setminus \{i\}}(t) - z_{N \setminus \{i\}}(t \setminus \{a_s\}) \\ &\geq \alpha(\text{pos}_i(a)) + z_{N \setminus \{i\}}(t) \\ &\quad - \alpha(\text{pos}_i(a_t)) - z_{N \setminus \{i\}}(t \setminus \{a_t\}) \\ &= z_N(t \cup \{a\}) - z_N(t) \end{aligned}$$

This completes the proof. □

3.4 Algorithm C (CC)

This algorithm, introduced in this paper, proceeds like Algorithm GM for Chamberlin-Courant's rule, but in each iteration it keeps up to d (partial) CC-representation functions Φ^S_{CC}, for distinct subsets S of alternatives (d is a parameter of the algorithm). In each iteration the algorithm extends each function Φ^S_{CC} by every possible alternative (obtaining $O(dm)$ new representation functions) and stores up to d of them, that obtain highest satisfaction.

3.5 Algorithm P (CC)

Algorithm P (position restriction) was introduced and studied by Skowron et al. [15]. The algorithm proceeds as follows. First, we consider a certain number x (specifically, $x = \lceil \frac{m\text{W}(K)}{K} \rceil$, where $\text{W}(x)$ is Lambert's W function, defined as the solution of equality $x = \text{W}(x)e^{\text{W}(x)}$). Then, the algorithm tries to greedily find a cover of as many agents as possible with K alternatives (an alternative is said to cover a given agent if this agent ranks this alternative among top x positions). Skowron et al. [15] have shown that for α^m_{Borda} this algorithm finds a solution that is at most $1 - \frac{2\text{W}(K)}{K}$ times worse than a perfect (possibly nonexistent) solution, where every agent is represented by his or her top-preferred alternative. The pseudocode for Algorithm P is presented in Figure 3.

3.6 Algorithm R (Monroe and CC)

Algorithm R (random sampling) is based on picking the set of winners randomly and matching them optimally to the agents. Skowron et al. [15] have shown that if one chooses a set S of K alternatives uniformly at random, then for α^m_{Borda}-Monroe, the expected satisfaction of $\alpha^m_{\text{Borda}}(\Phi^S_{\text{Monroe}})$ is $\frac{1}{2}(1 + \frac{K}{m} - \frac{K^2}{m^2 - m} + \frac{K^3}{m^3 - m^2}) - \epsilon$, and that one has to repeat this process $\frac{-512 \log(1-\lambda)}{K\epsilon^2}$ times, to reach probability λ of achieving this satisfaction. For example, for $\lambda = 0.99$ and $\epsilon = 0.1$ this algorithm would require to repeat the sampling process $340000/K$ times (each time executing

Alg.	Approx.	Runtime	Reference
A	$1-\frac{K-1}{2(m-1)}-\frac{H_K}{K}$	Kmn	Skowron et al. [15]
B	as in Alg. A	$Kmn+O(\Phi^S)$	(this paper)
C	as in Alg. A	$dKmn+dO(\Phi^S)$	(this paper)
GM	as in Alg. A for Borda PSF; $1-\frac{1}{e}$ for others	$KmO(\Phi^S)$	(this paper)
R	$\frac{1}{2}(1+\frac{K}{m}-\frac{K^2m-K^3}{m^3-m^2})$	$\frac{\lvert\log(1-\lambda)\rvert}{K\epsilon^2}O(\Phi^S)$	Skowron et al. [15]
P	$1-\frac{2\mathrm{W}(K)}{K}$	$nm\mathrm{W}(K)$	Skowron et al. [15]
GM	$1-\frac{1}{e}$	Kmn	Lu and Boutilier [5]
C	as in Alg. GM	$dKm(n+\log dm)$	(this paper)
R	$(1-\frac{1}{K+1})(1+\frac{1}{m})$	$\frac{\lvert\log(1-\lambda)\rvert}{\epsilon^2}n$	Oren [12]

Table 1: A summary of the algorithms studied in this paper. The top of the table regards algorithms for Monroe's rule and the bottom for Chamberlin-Courant's rule. In column "Approx." we give currently known approximation ratio for the algorithm under Borda PSF, on profiles with m candidates and where the goal is to select a committee of size K. Here, $O(\Phi^S)=O(n^2(K+\log n))$ is the complexity of finding a partial representation function with the algorithm of Betzler et al. [2].

a costly matching algorithm). This makes the algorithm impractical, especially for small instances (where K is low). Thus in our experimental evaluation we will consider the modification of the algorithm that repeats the sampling process only 100 times.

Oren [12] has shown an analogous result for the case of Chamberlin-Courant's rule.

3.7 Summary of the Algorithms

We summarize the algorithms that we use in Table 1. In particular, the table clearly shows that for the case of Monroe, Algorithms B and C are not much slower than Algorithm A but offer a chance of improved peformance. Algorithm GM is intuitively even more appealing, but achieves this at the cost of high time complexity. For the case of Chamberlin-Courant's rule, it is unclear which of the algorithms to expect to be superior. One of the main goals of this paper is to establish if either of the presented algorithms clearly dominates the others. Our implementations are available at `http://mimuw.edu.pl/~ps219737/monroe/experiments.tar.gz`.

4. EXPERIMENTAL DATA

We have considered both real-life preference-aggregation data and synthetic data, generated according to a number of election models.

4.1 Real-Life Data

We have used real-life data regarding people's preference on sushi types, movies, college courses, and competitors' performance in figure-skating competitions.

One of the major problems regarding real-life preference data is that either people express preferences over a very limited set of alternatives, or their preference orders are partial. To address the latter issue, for each such data set we complemented the partial orders to be total orders using the technique of Kamishima [4]. (The idea is to complete each preference order based on those reported preference orders that appear to be similar.)

Some of our data sets contain a single profile, whereas the others contain multiple profiles. When preparing data for a given number m of candidates and a given number n of voters from a given data set, we used the following method: We first uniformly at random chose a profile within the data set, and then we randomly selected n voters and m candidates. We used preference orders of these n voters restricted to these m candidates.

Sushi Preferneces. We used the set of preferences regarding sushi types collected by Kamishima [4].[2] Kamishima has collected two sets of preferences, which we call S1 and S2. Data set S1 contains complete rankings of 10 alternatives collected from 5000 voters. S2 contains partial rankings of 5000 voters over a set of 100 alternatives (each vote ranks 10 alternatives). We used Kamishima [4] technique to obtain total rankings.

Movie Preferences. Following Mattei et al. [8], we have used the NetFlix data set[3] of movie preferences (we call it MV). NetFlix data set contains ratings collected from about 480 thousand distinct users regarding 18 thousand movies. The users rated movies by giving them a score between 1 (bad) and 5 (good). The set contains about 100 million ratings. We have generated 50 profiles using the following method: For each profile we have randomly selected 300 movies, picked 10000 users that ranked the highest number of the selected movies, and for each user we have extended his or her ratings to a complete preference order using the method of Kamishima [4].

Course Preferences. Each year the students at the AGH University choose courses that they would like to attend. The students are offered a choice of six courses of which they have to attend three. Thus the students are asked to give an unordered set of their three top-preferred courses and a ranking of the remaining ones (in case too many students select a course, those with the highest GPA are enrolled and the remaining ones are moved to their less-preferred courses). In this data set, which we call CR, we have 120 voters (students) and 6 alternatives (courses). However, due to the nature of the data, instead of using Borda count PSF as the satisfaction measure, we have used the vector $(3,3,3,2,1,0)$. We made this data set publicly available under the url: `http://mimuw.edu.pl/~ps219737/monroe/registration.tar.gz`.

Figure Skating. This data set, which we call SK, contains preferences of the judges over the performances in a figure-skating competitions. The data set contains 48 profiles, each describing a single competition. Each profile contains preference orders of 9 judges over about 20 participants. The competitions include European skating championships, Olympics, World Junior, and World Championships, all from 1998[4]. (Note that while in figure skating judges provide numerical scores, this data set is preprocessed to contain preference orders.)

4.2 Synthetic Data

For our tests, we have also used profiles generated using three well-known distributions of preference orders.

Impartial Culture. Under impartial culture model of preferences (which we denote IC), for a given set A of alternatives, each voter's preference order is drawn uniformly at random from the set of all possible total orders over A. While not very realistic, profiles generated using impartial culture model are a standard testbed of election-related algorithms.

Polya-Eggenberger Urn Model. Following Walsh [16], we have used the Polya-Eggenberger urn model [1] (which we denote UR). In this model we generate votes as follows. We have a set A of m alternatives and an urn that initially contains all $m!$ preference

[2]The sushi data set is available under the following url: `http://www.kamishima.net/sushi/`

[3]http://www.netflixprize.com/

[4]This data set is available under the following url: `http://rangevoting.org/SkateData1998.txt`.

orders over A. To generate a vote, we simply randomly pick one from the urn (this is our generated vote), and then—to simulate correlation between voters—we return a copies of this vote to the urn. When generating an election with m candidates using the urn model, we have set the parameter a so that $\frac{a}{m!} = 0.05$ (Walsh [16] calls this parameter b; we mention that other authors use much higher values of b but we felt that too high a value of b leads to a much too strong a correlation between votes).

Generalized Mallow's Model. We refer to this data set as ML. Let $\succ$ and $\succ'$ be two preference orders over some alternative set A. Kendal-Tau distance between $\succ$ and $\succ'$, denoted $d_K(\succ, \succ')$, is defined as the number of pairs of candidates $x, y \in A$ such that either $x \succ y \wedge y \succ' x$ or $y \succ x \wedge x \succ' y$.

Under Mallow's distribution of preferences [7] we are given two parameters: A *center* preference order $\succ$ and a number ϕ between 0 and 1. The model says that the probability of generating preference order $\succ'$ is proportional to the value $\phi^{d_K(\succ,\succ')}$. To generate preference orders following Mallow's distribution, we use the algorithm given by Lu and Boutilier [6].

In our experiments, we have used a mixture of Mallow's models. Let A be a set of alternatives and let a be a positive integer. This mixture model is parametrized by three vectors, $\Lambda = (\lambda_1, \ldots, \lambda_a)$ (where each λ_i, $1 \leq i \leq a$ is between 0 and 1, and $\sum_{i=1}^{a} \lambda_1 = 1$), $\Phi = (\phi_1, \ldots, \phi_a)$ (where each ϕ_i, $1 \leq i \leq a$, is a number between 0 and 1), and $\Pi = (\succ_1, \ldots, \succ_a)$ (where each $\succ_i$, $1 \leq i \leq a$, is a preference order over A). To generate a vote, we pick a random integer i, $1 \leq i \leq a$ (each i is chosen with probability λ_i), and then generate the vote using Mallow's model with parameters $(\succ_i, \phi_i)$.

For our experiments we have used $a = 5$, and we have generated vectors Λ, Φ, and Π uniformly at random.

5. EXPERIMENTS

In this section we present the results of the evaluation of algorithms from Section 3 on the data sets from Section 4. In all cases, except for the college courses data set, we have used Borda PSF to measure voter satisfaction. For the case of the courses data set, we have used vector $(3, 3, 3, 2, 1, 0)$.

We have conducted three sets of experiments. First, we have tested all our algorithms on relatively small elections (up to 10 candidates, up to 100 agents). In this case we were able to compare our algorithms' solutions with the optimal ones. (To obtain the optimal solutions we were using the ILP formulations and GLPK's ILP solver.) Thus we report the quality of our algorithms as the average of fractions C/C_{opt}, where C is the satisfaction obtained by a respective algorithm and C_{opt} is the satisfaction in the optimal solution. For each algorithm and data set, we also report the average fraction C/C_{ideal}, where C_{ideal} is the satisfaction that the voters would have obtained if each of them were matched to his or her most preferred alternative. In our further experiments, where we consider larger elections, we were not able to compute optimal solutions, but fraction C/C_{ideal} gives a lower bound for C/C_{opt}. We report this value for small elections so that we see an example of relation between C/C_{opt} and C/C_{ideal} and so that we can compare the results for small elections with the results for the larger ones. Further, for the case of Borda PSF the C/C_{ideal} fraction has a very natural interpretation: If its value is α (for a given solution), then, on the average, in this solution each voter is matched to an alternative that he or she prefers to $(m-1)\alpha$ alternatives.

In our second set of experiments we have run our algorithms on large elections (thousands of agents, hundreds of alternatives), coming either from the NetFlix data set or generated by us using one of our models. Here we reported the average fraction C/C_{ideal} only. We have analyzed the quality of the solutions as a function of the number of agents, the number of candidates, and the relative number of winners (fraction K/m). (This last set of results is particularly interesting because in addition to measuring the quality of our algorithms, it allows one to asses the size of a committee one should seek if a given average agent satisfaction is to be obtained).

In the third set of experiments we have measured running times of our algorithms and of the ILP solver.

Table 2: The average quality of the algorithms compared with the optimal solution (C/C_{opt}) for the small instances of data and for $K = 3$.

	Monroe					CC			
	A	B	C	GM	R	C	GM	P	R
S1	0.94	0.99	≈ 1.0	0.99	0.99	1.0	≈ 1.0	0.99	0.99
S2	0.95	0.99	1.0	≈ 1.0	0.99	1.0	≈ 1.0	0.98	0.99
MV	0.96	≈ 1.0	1.0	≈ 1.0	0.98	1.0	≈ 1.0	0.96	≈ 1.0
CR	0.98	0.99	1.0	≈ 1.0	0.99	1.0	≈ 1.0	1.0	≈ 1.0
SK	0.99	≈ 1.0	1.0	≈ 1.0	0.94	1.0	≈ 1.0	0.85	0.99
IC	0.94	0.99	≈ 1.0	0.99	0.99	1.0	≈ 1.0	0.99	0.99
ML	0.94	0.99	1.0	0.99	0.99	1.0	≈ 1.0	0.99	0.99
UR	0.95	0.99	≈ 1.0	0.99	0.99	1.0	0.99	0.97	0.99

Table 3: The average quality of the algorithms compared with the simple lower bound (C/C_{ideal}) for the small instances of data and for $K = 3$.

	Monroe					CC			
	A	B	C	GM	R	C	GM	P	R
S1	0.85	0.89	0.9	0.89	0.89	0.92	0.89	0.91	0.92
S2	0.85	0.89	0.89	0.89	0.89	0.93	0.9	0.91	0.92
MV	0.88	0.92	0.92	0.92	0.91	0.97	0.92	0.93	0.97
CR	0.94	0.97	0.96	0.96	0.96	0.97	0.97	0.97	0.97
SK	0.96	0.96	0.97	0.97	0.91	1.0	0.97	0.82	0.99
IC	0.8	0.84	0.85	0.84	0.84	0.85	0.83	0.84	0.85
ML	0.83	0.88	0.88	0.9	0.88	0.92	0.90	0.89	0.94
UR	0.8	0.85	0.86	0.87	0.85	0.9	0.87	0.87	0.89

5.1 Evaluation on Small Instances

We now present the results of our experiments on small elections. For each data set, we generated elections with the number of agents $n = 100$ ($n = 9$ for data set SK because there are only 9 voters there) and with the number of alternatives $m = 10$ ($m = 6$ for data set CR because there are only 6 alternatives there) using the method described in Section 4.1 for the real-life data sets, and in the natural obvious way for synthetic data. For each algorithm and for each data set we ran 500 experiments on different instances for $K = 3$ (for the CR data set we used $K = 2$) and 500 experiments for $K = 6$ (for CR we set $K = 4$). For Algorithms C we set the parameter $d = 15$. The results (average fractions C/C_{opt} and C/C_{ideal}) for $K = 3$ are given in Tables 2 and Tables 3. We omit the results for $K = 6$ due to space constraints, but we mention they are almost identical (we can provide all our omitted numerical results upon request). For each experiment in this section we also computed the standard deviation; it was always on the order of 0.01. The results lead to the following conclusions:

1. Even Algorithm A obtains very good results, but nonetheless Algorithms B and C improve noticeably upon Algorithm A. In particular, Algorithm C (for $d = 15$) obtains the highest satisfaction on all data sets and in almost all cases was able to find an optimal solution.
2. Algorithm R gives slightly worse solutions than Algorithm C.
3. The quality of the algorithms does not depend on the data set used for verification (the only exception is Algorithm R for Monroe's rule on data set SK; however SK has only 9 voters so it can be viewed as a border case).

5.2 Evaluation on Larger Instances

For experiments on larger instances we needed data sets with at least $n = 10000$ agents. Thus we used the NetFlix data set and synthetic data. (Additionally, we run the subset of experiments (for $n \leq 5000$) also for the S2 data set.) For Monroe's rule we present results for Algorithm A, Algorithm C, and Algorithm R, and for Chamberlin-Courant's rule we present results for Algorithm C and Algorithm R. We limit the set of algorithms due to space constraints. For Monroe we chose Algorithm A because it is the simplest and the fastest one, Algorithm C because it is the best generalization of Algorithm A that we were able to run in reasonable time, and Algorithm R to compare a randomized algorithm to deterministic ones. For Chamberlin-Courant's rule we chose Algorithm C because it is, intuitively, the best one, and we chose Algorithm R for the same reason as in the case of Monroe. Further, we present results for the NetFlix data set and for the urn model only. Again, we do so due to space constraints, and we chose these data sets because the urn model results turned out to be the worst ones among the synthetic data sets, and the NetFlix data set is our only large real-life data set. (All the omitted results are available upon request.)

First, for each data set and for each algorithm we fixed the value of m and K and for each n ranging from 1000 to 10000 with the step of 1000 we run 50 experiments. We repeated this procedure for 4 different combinations of m and K: ($m = 10$, $K = 3$), ($m = 10$, $K = 6$), ($m = 100$, $K = 30$) and ($m = 100$, $K = 60$). We measured the statistical correlation between the number of voters and the quality of the algorithms C/C_{ideal}. The ANOVA test in most cases showed that there is no such correlation. The only exception was S2 data set, for which we obtained an almost negligible correlation. For example, for ($m = 10, K = 3$) Algorithm C under data set S2 for Monroe's rule for $n = 5000$ gave $C/C_{\text{ideal}} = 0.88$, while for $n = 100$ (in the previous section) we got $C/C_{\text{ideal}} = 0.89$. Thus we conclude that in practice the number of agents has almost no influence on the quality of the results provided by our algorithms.

Next, we fixed the number of voters $n = 1000$ and the ratio $K/m = 0.3$, and for each m ranging from 30 to 300 with the step of 30 (naturally, as m changed, so did K to maintain the ratio K/m), we run 50 experiments. We repeated this procedure for $K/m = 0.6$. The relation between m and C/C_{ideal} for MV and UR, under both Monroe's rule and Chamberlin-Courant's rule, is given in Figures 4 and 5 (the results for $K/m = 0.6$ look similar).

Finally, we fixed $n = 1000$ and $m = 100$, and for each K/m ranging from 0.1 and 0.5 with the step of 0.1 we run 50 experiments. The relation between the ratio K/m and the quality C/C_{ideal} is presented in Figures 6 and 7.

For the case of Chamberlin-Courant's rule increasing the size of the committee we elect improves agent satisfaction: Since there are no constraints on the number of agents matched to a given alternative, larger committees mean more opportunities to satisfy the agents. For Monroe, larger committees may lead to lower total satisfaction. This happens if many agents like a particular alternative a lot, but only some of them can be matched to this alternative and others have to be matched to their less-preferred ones. Nonetheless, we see that Algorithm C achieves $C/C_{\text{ideal}} = 0.925$ even for $K/m = 0.5$ for the NetFlix data set.

Our conclusions from these experiments are the following. For Monroe's rule, even Algorithm A achieves very good results. However, Algorithm C consistently achieves better ones (indeed, almost perfect ones). Randomized algorithms consistently do worse than our deterministic ones.

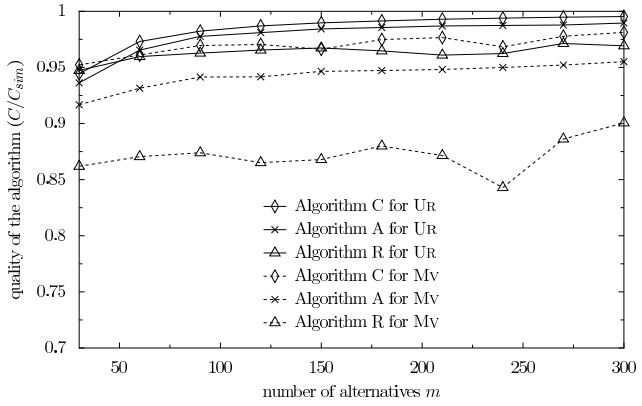

Figure 4: The relation between the number of alternatives m and the quality of the algorithms C/C_{ideal} for the Monroe's rule.

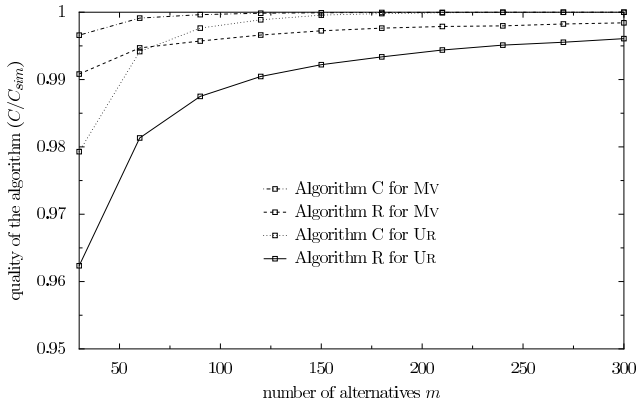

Figure 5: The relation between the number of alternatives m and the quality of the algorithms C/C_{ideal} for the Chamberlin-Courant's rule.

5.3 Running time

In our final set of experiments we have measured running times of our algorithms on the data set MV. We have used a machine with Intel Pentium Dual T2310 1.46GHz processor and 1.5GB of RAM. In Figure 8 we show the running time of GLPK ILP solver for Monroe's and for Chamberlin-Courant's rules. These running times are already large for small instances and they are increasing exponentially with the number of voters. For Monroe's rule, even for $K = 9, m = 30, n = 100$ some of the experiments timed out after 1 hour, and for $K = 9, m = 30, n = 200$ none of the experiments finished within one day. Thus we conclude that the real application of the ILP algorithm is very limited. Example running times of the other algorithms for some combinations of n, m, and K are presented in Table 4.

6. CONCLUSIONS

We have provided experimental evaluation of a number of algorithms (both known ones and their extensions) for computing the winners under Monroe's rule and under Chamberlin-Courant's rule. While finding winners under these rules is NP-hard [2,5,14], it turned out that in practice we can obtain very high quality solutions using simple algorithms. Indeed, both for Monroe's rule and for Chamberlin-Courant's rule we recommend using Algorithm C (or Algorithm A on very large Monroe elections). We believe that our

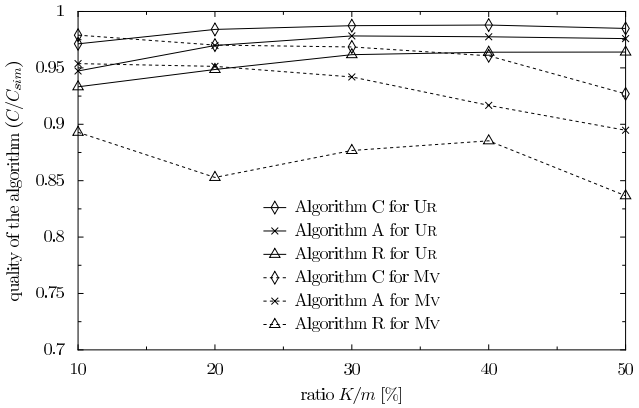

Figure 6: The relation between the ratio K/m and the quality of the algorithms C/C_{ideal} for the Monroe's rule.

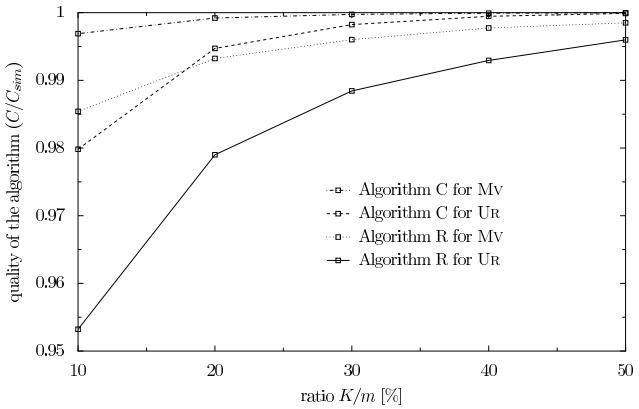

Figure 7: The relation between the ratio K/m and the quality of the algorithms C/C_{ideal} for the Chamberlin-Courant's rule.

results mean that (approximations of) Monroe's and Chamberlin-Courant's rules can be used in practice.

Acknowledgements The authors were supported in part by AGH Univ. grant 11.11.120.865, by the Foundation for Polish Science's Homing/Powroty program, by Poland's National Science Center grant DEC-2011/03/B/ST6/01393, and by EU's Human Capital Program "National PhD Programme in Mathematical Sciences" carried out at the University of Warsaw.

7. REFERENCES

[1] S. Berg. Paradox of voting under an urn model: The effect of homogeneity. *Public Choice*, 47:377–387, 1985.

[2] N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. Technical report, U. of Auckland, November 2011.

[3] B. Chamberlin and P. Courant. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *American Political Science Review*, 77(3):718–733, 1983.

[4] T. Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of KDD-03*, pages 583–588, 2003.

[5] T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of IJCAI-2011*, pages 280–286, 2011.

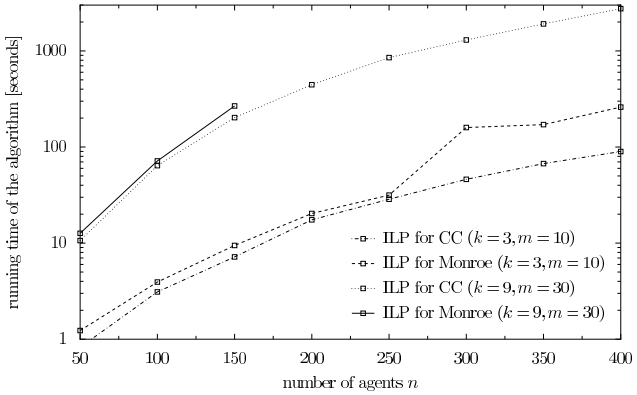

Figure 8: The running time of the standard ILP solver for Monroe's and Chamberlin-Courant's rules. For Monroe's rule, for $K = 9, m = 30$ and for $n \geq 200$ none of the single algorithm execution finished within 1 day.

Table 4: Example running times of the algorithms [in seconds].

		$m = 100, K = 30$			$m = 100, K = 60$		
	$n =$	2000	6000	10000	2000	6000	10000
Monroe	A	0.5	1.6	2.8	0.9	2.8	4.9
	B	0.8	4	9.5	1.7	8	18
	C	38	140	299	64	221	419
	GM	343	2172	5313	929	5107	13420
	R	41	329	830	88	608	1661
CC	C	4.3	11	19	7.5	19	31
	GM	0.06	0.2	0.4	0.09	0.3	0.7
	P	0.03	0.1	0.26	0.03	0.1	0.2
	R	0.06	0.24	0.45	0.1	0.4	0.8

[6] T. Lu and C. Boutilier. Learning Mallows models with pairwise preferences. In *Proceedings of ICML-11*, pages 145–152, June 2011.

[7] C. L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1-2):114–130, June 1957.

[8] N. Mattei, J. Forshee, and J. Goldsmith. An empirical study of voting rules and manipulation with large datasets. In *COMSOC*, 2012.

[9] R. Meir, A. Procaccia, J. Rosenschein, and A. Zohar. The complexity of strategic behavior in multi-winner elections. *JAIR*, 33:149–178, 2008.

[10] B. Monroe. Fully proportional representation. *American Political Science Review*, 89(4):925–940, 1995.

[11] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[12] J. Oren. Personal communication, 2012.

[13] R. Potthoff and S. Brams. Proportional representation: Broadening the options. *Journal of Theoretical Politics*, 10(2):147–178, 1998.

[14] A. Procaccia, J. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008.

[15] P. Skowron, P. Faliszewski, and A. Slinko. Fully proportional representation as resource allocation: Approximability results. Technical Report arXiv:0809.4484 [cs.GT], arXiv.org, Jan. 2013.

[16] T. Walsh. Where are the hard manipulation problems? *JAIR*, 42:1–29, 2011.